Overview of Tools in the Hadoop Ecosystem

Lecture BigData Analytics

Julian M. Kunkel

julian.kunkel@googlemail.com

University of Hamburg / German Climate Computing Center (DKRZ)

2017-12-22

Disclaimer: Big Data software is constantly updated, code samples may be outdated.
Outline

1. Hadoop Ecosystem
2. User/Admin Interfaces
3. Workflows
4. SQL Tools
5. Other BigData Tools
6. Machine Learning
7. Summary
Hortonworks

- Additionally: Hortonworks offers support, service
- Build with open-source
Cloudera Enterprise Hadoop Ecosystem [25]

- Cloudera offers support, services and tools around Hadoop
- Unified architecture: common infrastructure and data pool for tools
- Build with open-source tools, some own tools for management, encryption

Source: [26]
Supporting Tools\(^1\)

- **Ambari**: A Tool for Managing Hadoop Clusters
- **Hue**: Manage „BigData“ projects in a browser
- **ZooKeeper**: coordination/configuration service for services
- **Sqoop**: ETL between HDFS and structured data stores
- **Oozie**: Workflow scheduler (schedules/triggers workflows)
- **Falcon**: Data governance engine for data pipelines
- **Flume**: collecting, aggregating and moving large streaming event data
- **Kafka**: publish-subscribe distributed messaging system
- **Knox**: REST API gateway (for all services)
- **Ranger**: Integrate ACL permissions into Hadoop (ecosystem)
- **Slider**: YARN application supporting monitoring and dynamic scaling of non-YARN apps

\(^1\)https://hadoop.apache.org/
Ambari: A Tool for Managing Hadoop Clusters

- Convenient tool managing 10+ Apache tools
- Supports installation and management
  - Dealing with data dependencies
  - Service startup
  - Monitoring of health and performance
  - (Re)configuration of services
Management with Ambari: Dashboard

Screenshot from the WR-cluster Ambari
Management with Ambari: Configuration

- Restart Required: 1 Component on 1 Host

- Group: HDFS Default (5)
- Manage Config Groups:
  - V2 admin 2 months ago
  - Y1 admin 2 months ago
  - Current admin authored on Tue, Jul 07, 2015 19:05

- NameNode:
  - NameNode hosts: abu1.cluster
  - NameNode directories: /tmp/hadoop/hdfs/namenode, /bigdata/hdfs/namenode
  - NameNode Java heap size: 95744 MB
  - NameNode new generation size: 23936 MB
  - NameNode maximum new generation size: 23936 MB
  - NameNode permanent generation size: 128 MB
  - NameNode maximum permanent generation size: 128 MB
Knox: Security for Hadoop [22]

- **REST API Gateway for Hadoop ecosystem services**
  - Supports: HDFS, Hcatalog, HBase, Oozie, Hive, Yarn, Storm
  - Supports multiple clusters
- **Provides authentication, federation/SSO, authorization, auditing**
- **Enhances security providing central control and protection**
  - SSL encryption
  - Authentication: LDAP, Active Directory, Kerberos
  - Authorization: ACL’s (user, group, IP) on service level

Source: [22]
Example Accesses via the REST API [22]

List a HDFS directory

```
curl -i -k -u guest:guest-password -X GET
  ↪ 'https://localhost:8443.gateway/sandbox/webhdfs/v1/?op=LISTSTATUS'
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 450
Server: Jetty(6.1.26)

{"FileStatuses":{"FileStatus":[
  "accessTime":0,"blockSize":0,"group":"hdfs","length":0,
  "modificationTime":1350595859762,"owner":"hdfs","pathSuffix":"apps",
  "permission":"755","replication":0,"type":"DIRECTORY"},
  "accessTime":0,"blockSize":0,"group":"mapred","length":0,
  "modificationTime":1350595874024,"owner":"mapred","pathSuffix":"mapred",
  "permission":"755","replication":0,"type":"DIRECTORY"},

]}}
```
Hue [12]: Lightweight Web Server for Hadoop

- Manage BigData projects in a browser
- Supports: Hadoop ecosystem
  - HDFS, Pig, Sqoop, Hive, Impala, MapReduce, Spark, ...

Features

- Data upload/download
- Management of HCatalog tables
- Query editor (Hive, Pig, Impala)
- Starting and monitoring of jobs
Hue: Lightweight Web Server for Hadoop

Monitoring Oozie Workflows (Live system on gethue.com)
Hue: Lightweight Web Server for Hadoop

File browser (Live system on gethue.com)
Hue: Lightweight Web Server for Hadoop

Query editor (Live system on gethue.com)
Hue: Lightweight Web Server for Hadoop

Visualizing query results in diagrams (Live system on gethue.com)
Zeppelin [39]

- Web-based notebook for interactive data analytics
  - Add code snippets
  - Arrange them
  - Execute them
  - Visualizes results

- Supports Spark, Scala, Pig, SQL, Python, R, Hive, Shell, ...

- Collaborative environment for multiple users

- Can export paragraph links for embedding into a webpage
Zeppelin

Zeppelin Tutorial

```scala
val bankText = sc.parallelize(
  IOUtils.toLines(new URL("https://s3.amazonaws.com/apache-zeppelin/tutorial/bank/bank.csv"),
  Charsets.UTF_8).split('"'))

case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)

val bank = bankText.map(s => s.split("").filter(s => s(0) != "\"age\"").map(
  s => Bank(s(0).toInt,
    s(1).replaceAll("",""),
    s(2).replaceAll("",""),
    s(3).replaceAll("",""),
    s(4).replaceAll("",""))).toInt
)
bankDF.registerTempTable("bank")

Too many open files
```

```sql
select age, count(1) value
from bank
where marital="single,single\|divorced\|married"
geroup by age
order by age
```

```sql
select age, count(1) value
from bank
where age = $\{\text{maxAge-30}\}$
geroup by age
order by age
```
1 Hadoop Ecosystem
2 User/Admin Interfaces
3 Workflows
4 SQL Tools
5 Other BigData Tools
6 Machine Learning
7 Summary
Oozie [15, 16]

- Scalable, reliable and extensible workflow scheduler
- Jobs are DAGs of actions specified in XML workflows
- Actions: Map-reduce, Pig, Hive, Sqoop, Spark, Shell actions
- Workflows can be parameterized
  - Triggers notifications via HTTP GET upon start/end of a node/job
  - Automatic user-retry to repeat actions when fixable errors occur
  - Monitors a few runtime metrics upon execution
- Interfaces: command line tools, web-service and Java APIs
- Integrates with HCatalog
- Coordinator jobs trigger start of jobs
  - By time schedules
  - When data becomes available
    - Requires polling of HDFS (1-10 min intervals)
    - With HCatalog’s publish-subscribe, jobs can be started immediately
  - Can record events for service level agreement
Workflows [16]

- A workflow application is a ZIP file to be uploaded
  - Includes workflow definition and coordinator job
  - Bundles scripts, JARs, libraries needed for execution

- Workflow definition is a DAG with control flow and action nodes
  - Control flow: start, end, decision, fork, join
  - Action nodes: whatever to execute

- Variables/Parameters
  - Default values can be defined in a config-default.xml in the ZIP

- Expression language functions help in parameterization
  - Basic functions: `timestamp()`, `trim()`, `concat(s1, s2)`
  - Workflow functions: `wf:errorCode(< action node >)`
  - Action specific functions:
    - `hadoop:counters("mr-node")["FileSystemCounters"]["FILE_BYTES_READ"]`

- Coordinator job is also an XML file

---

3 They are used with with `${NAME/FUNCTION}`, e.g., `${timestamp()}`
Coordinator Jobs [17]

App which periodically starts a workflow (every 60 min)

```xml
<coordinator-app name="MY_APP" frequency="60" start="2009-01-01T05:00Z" end="2009-01-01T06:00Z" timezone="UTC"
    xmlns="uri:oozie:coordinator:0.1">
  <action>
    <workflow> <!-- here the workflow is not further defined -->
      <app-path>hdfs://localhost:9000/tmp/workflows</app-path>
    </workflow>
  </action>
</coordinator-app>
```

Every 24h check if dependencies for a workflow are met, then run it

```xml
<coordinator-app name="MY_APP" frequency="1440" start="2009-02-01T00:00Z" end="2009-02-07T00:00Z" ...
    xmlns="uri:oozie:coordinator:0.1">
  <datasets> <!-- define a dataset, that is checked for existence -->
    <dataset name="input1" frequency="60" initial-instance="2009-01-01T00:00Z" timezone="UTC">
      <uri-template>hdfs://localhost:9000/tmp/revenue_feed/${YEAR}/${MONTH}/${DAY}/${HOUR}</uri-template>
    </dataset>
  </datasets>
  <input-events> <!-- we depend on the last 24 hours input data -->
    <data-in name="coordInput1" dataset="input1">
      <start-instance>${coord:current(-23)}</start-instance>
      <end-instance>${coord:current(0)}</end-instance>
    </data-in>
  </input-events>
  <action>
    <workflow>
      <app-path>hdfs://localhost:9000/tmp/workflows</app-path>
    </workflow>
  </action>
</coordinator-app>
```
Example Oozie Workflow [13]

Three actions: Execute pig script, concatenate reducer files, upload files remotely via ssh

```xml
<workflow-app xmlns="uri:oozie:workflow:0.2" name="sample-wf">
  <start to="pig" />
  <action name="pig">
    <pig>
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <prepare><delete path="${output}"/></prepare>
      <configuration>
        <property>
          <name>mapred.job.queue.name</name>
          <value>${queueName}</value>
        </property>
        <property>
          <name>mapreduce.fileoutputcommitter.marksuccessfuljobs</name>
          <value>true</value>
        </property>
      </configuration>
      <script>${nameNode}/projects/bootcamp/workflow/script.pig</script>
      <param>input=${input}</param>
      <param>output=${output}</param>
      <file>lib/dependent.jar</file>
    </pig>
    <ok to="concatenator" />
    <error to="fail" />
  </action>

  <action name="fileupload">
    <ssh>
      <host>localhost</host>
      <command>/tmp/fileupload.sh</command>
      <args>${nameNode}/projects/bootcamp/concat/data-${fileTimestamp}.csv</args>
      <capture-output /></ssh>
    <ok to="fileUploadDecision" />
    <error to="fail" />
  </action>

  <decision name="fileUploadDecision">
    <switch>
      <case to="end">${wf:actionData('fileupload')['output'] == '0'}</case>
      <default to="fail" />
    </switch>
  </decision>

  <kill name="fail">
    <message>Workflow failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
  </kill>

  <end name="end" />
</workflow-app>
```
Falcon [11,13]

- Feed (data set) management and processing system
- Simplifies dealing with many Oozie jobs
- Supports data governance
  - Define and run data pipelines (management policies)
  - Monitor data pipelines
  - Trace pipelines to identify dependencies and perform audits
- Data model defines entities describing policies and pipelines
  - Clusters define resources and interfaces to use
  - Feeds define frequency, data retention, input, outputs, retry and use clusters (multiple for replication)
  - Process: processing task, i.e., Oozie workflow, Hive or Pig script
- Features
  - Supports reuse of entities for different workflows
  - Enables replication across clusters and data archival
  - Supports HCatalog
  - Notification of users upon availability of feed groups
Falcon: High-level Architecture

Source: [11]
Falcon: Example Pipeline

Source: [11]
Falcon: Example Process Definition [11, 14]

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Sample process. Runs at 6th hour every day. Input: last day hourly data. Output: for yesterday -->
<process name="SampleProcess">
  <cluster name="wr" />
  <frequency>days(1)</frequency>

  <validity start="2015-04-03T06:00Z" end="2022-12-30T00:00Z" timezone="UTC" />

  <inputs>
    <input name="input" feed="SampleInput" start="yesterday(0,0)" end="today(-1,0)" />
  </inputs>

  <outputs>
    <output name="output" feed="SampleOutput" instance="yesterday(0,0)" />
  </outputs>

  <properties>
    <property name="queueName" value="reports" />
    <property name="ssh.host" value="host.com" />
    <property name="fileTimestamp" value="${coord:formatTime(coord:nominalTime(), 'yyyy-MM-dd')}" />
  </properties>

  <workflow engine="oozie" path="/projects/bootcamp/workflow" />

  <retry policy="backoff" delay="minutes(5)" attempts="3" />

  <!-- How to check and handle late arrival of input data-->
  <late-process policy="exp-backoff" delay="hours(1)">
    <late-input input="input" workflow-path="/projects/bootcamp/workflow/lateinput" />
  </late-process>
</process>
```
Atlas [23]

- A framework for platform-agnostic data governance
- Exchange metadata with other tools
- Audit operations, explore history of data and metadata
- Support lifecycle management workflows built with Falcon
- Support Ranger access control (ACL’s)

Source: [23]
Sqoop [18, 19]

- Transfers bulk data between Hadoop and RDBMS, either
  - One/multiple tables (preserving their schema)
  - Results of a free-form SQL query
- Uses MapReduce to execute import/export jobs
  - Parallelism is based on splitting one column’s value
- Validate data transfer (comparing row counts) for full tables
- Save jobs for repeated invocation
- Main command line tool sqoop, more specific tools sqoop*
Features [19]

Import Features

- Incremental import (scan and add only newer rows)
- File formats: CSV, SequenceFiles, Avro, Parquet
  - Compression support
- Outsource large BLOBS/TEXT into additional files
- Import into Hive (and HBase)
- Can create the table schema in HCatalog automatically
  - With HCatalog, only CSV can be imported

Export Features

- Bulk insert: 100 records per statement
- Periodic commit after 100 statements
**Import Process [19]**

- Read the schema of the source table
- Create a Java class representing a row of the table
  - This class can be used later to work with the data
- Start MapReduce to load data parallel into multiple files
  - The number of mappers can be configured
  - Mappers work on different values of the splitting column
  - The default splitting column is the primary key
    - Determines min and max value of the key
    - Distributes fixed chunks to mappers
- Output status information to the MapReduce job tracker
Example Imports [19]

1 # Import columns from "foo" into HDFS to /home/x/foo (table name is appended)
2 # When not specifying any columns, all columns will be imported.
3 $ sqoop import --connect jdbc:mysql://localhost/db --username foo --table TEST --columns
   "matrikel,name" --warehouse-dir /home/x --validate

4 # We’ll use a free-form query, it is parallelized on the split-by column
5 # The value is set into the magic $CONDITIONS variable
6 $ sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id == b.id) WHERE
   $CONDITIONS' --split-by a.id --target-dir /user/foo/joinresults

8 # To create the HCatalog table use --hcatalog-table or --hive-import
8 # See [19] for details of the available options
Slider [20]

- Is a YARN application that manages non-YARN apps on a cluster
- Utilize YARN for resource management
- Enables installation, execution, monitoring and dynamic scaling
- Command line tool slider
- Apps are installed and run from a package
  - Tarball with well-defined structure [21]
  - Scripts for installing, starting, status, ...
- Example packages: jmemcached, HBase
- Slider is currently extended to deploy Docker images (Tech preview)
<table>
<thead>
<tr>
<th></th>
<th>Hadoop Ecosystem</th>
<th>User/Admin Interfaces</th>
<th>Workflows</th>
<th>SQL Tools</th>
<th>Other BigData Tools</th>
<th>Machine Learning</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hadoop Ecosystem</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>User/Admin Interfaces</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Workflows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SQL Tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Other BigData Tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Machine Learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Summary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Drill [10, 29, 30]

- Software framework for data-intensive distributed applications
- Data model: relational (ANSI SQL !) + schema-free JSON
- Analyse data in-situ without data movement
  - Execute one query against multiple NoSQL datastores
  - Datastores: HBase, MongoDB, HDFS, S3, Swift, local files
- Features
  - REST APIs
  - Columnar execution engine supporting complex data
  - Locality-aware execution
  - Cost-based optimizer pushing processing into datastore
  - Runtime compilation of queries

```sql
1 # Different datastores, localstorage, mongodb and s3
2 SELECT * FROM dfs.root.'/logs';
3 SELECT country, count(*) FROM mongodb.web.users GROUP BY country;
4 SELECT timestamp FROM s3.root.'users.json' WHERE user_id = 'max';
5
6 # Query JSON: access the first students age from private data (a map)
7 SELECT student[0].private.AGE, FROM dfs.'students.json';
```
Cloudera Impala [25, 26]

- Enterprise analytic database
  - Utilizes HDFS, HBase and Amazon S3
  - Based on Google Dremel like Apache Drill
- Written in C++, Java
- Massively-parallel SQL engine
  - Supports HiveQL and subset of ANSI-92 SQL
- Uses LLVM to generate efficient code for queries
Apache Metamodel [43]

- Provides a Java based SQL-alike interface to various data sources
  - CSV, SQL dbs, JSON, HBase, MongoDB

### Query [43]

```java
DataContext dataContext = DataContextFactory.create[TypeOfDatastore](...);
DataSet dataSet = dataContext.query()
                   .from("libraries").select("name").where("language").eq("Java").and("enhances_data_access").eq(true).execute();
```

### Update [43]

```java
dataContext.executeUpdate(new UpdateScript() {
    public void run(UpdateCallback callback) {
        // CREATE a table
        Table table = callback.createTable("contributors")
                      .withColumn("id").ofType(INTEGER).withColumn("name").ofType(VARCHAR).execute();

        // INSERT INTO table
        callback.insertInto(table).value("id", 1).value("name", "John Doe").execute();
        callback.insertInto(table).value("name", "Jane D.").execute();

        // UPDATE table
        callback.update(table).value("name","Jane Doe").where("id").eq(2).execute();

        // DELETE FROM table
        callback.deleteFrom(table).where("id").eq(1).execute();
    }
});
```
1. Hadoop Ecosystem
2. User/Admin Interfaces
3. Workflows
4. SQL Tools
5. Other BigData Tools
6. Machine Learning
7. Summary
Kafka [41]

- **Publish-subscribe** distributed messaging system
  - Producer publishes message for a given **topic**
  - Consumer subscribes to topic and receives data
  - Simple: Consumer has to remember its read position (**offset**)
- A data source for Storm, HBase, Spark, ...
- Use cases – support data ingestion:
  - GPS data from truck fleet, sensor data
  - Error logs from cluster nodes, web server activity
- Features
  - Parallel, fault-tolerant server system (a server is called **broker**)

Source: [42]
Solr [10, 31]

- Full-text search and indexing platform
- REST API: index documents and query via HTTP
  - Query response in JSON, XML, CSV, binary
- Features
  - Data can be stored on HDFS
  - High-availability, scalable and fault tolerant
  - Distributed search
  - Faceted classification: organize knowledge into a systematic order using (general or subject-specific) semantic categories that can be combined for a full classification entry [10]
  - Geo-spatial search
  - Caching of queries, filters and documents
- Uses lucene library for search
- Very similar: Elasticsearch [33], http://solr-vs-elasticsearch.com/
Example Query [32]

Identifying available facets terms and number of docs for each

```bash
1 curl http://localhost:8983/solr/gettingstarted/select?wt=json&indent=true&q=*:*&rows=0&facet=true&facet.field=manu_id_s
```

Response

```json
{
  "responseHeader":{
    "status":0,
    "QTime":3,
    "params":{ /* Parameters of the query */
      "facet":"true", "indent":"true", "q":"*:*", "facet.field":"manu_id_s", "wt":"json",
      "rows":"0"},
  "response":{"numFound":2990,"start":0,"docs":[]}, /* number of documents found */
  "facet_counts":{
    "facet_queries":{},
    "facet_fields":{ /* the available facets and number of documents */
      "manu_id_s":{"corsair":3, "belkin":2, "canon":2, "apple":1, "asus":1, "ati":1, "boa":1, "dell":1, "eu":1, "maxtor":1,
      "nor":1, "uk":1, "viewsonic":1, "samsung":0}},
    "facet_dates":{},
    "facet_ranges":{},
    "facet_intervals":{}
  }
}
```
<table>
<thead>
<tr>
<th></th>
<th>Hadoop Ecosystem</th>
<th>User/Admin Interfaces</th>
<th>Workflows</th>
<th>SQL Tools</th>
<th>Other BigData Tools</th>
<th>Machine Learning</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>Machine Learning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Mahout [34]

- Framework for scalable machine learning
  - Collaborative filtering
  - Classification
  - Clustering
  - Dimensionality reduction
  - Recommender
    - history: user purchases + all purchases ⇒ recommendations (user)

- Computation on Spark, MapReduce, H2O engines [36]
  - Can also use a single machine without Hadoop
  - Algorithm availability depends on the backend

- Bindings for Scala language [35]
  - Provide distributed BLAS, Row Matrix (DRM)
  - R-like DSL embedded in Scala
  - Algebraic optimizer
Recommender Architecture

1. Collect user interactions $n \times (\text{user-id, item-id})$
2. Learning:
   1. Itemsimilarity creates item, list-of-similar-items
   2. Store those tuples in the search engine
3. Query search engine with $n$ latest user interactions
4. If they occur in the list-of-similar-items, recommend item

Source: [36]
TensorFlow [44]

- Platform independent opens-source library for machine learning
- GPU accelerated
- Developed by GoogleBrain
- Tensor: Multidimensional array with rank dimensions
- Workflow
  - Define tensors (or placeholders)
  - Build execution graph based on equations and optimizers (math)
  - Start execution (bind dummy variables)
- Big community
- PyTorch alternative from Facebook with dynamic graph creation
# Here build a neural network of a linear classifier to estimate classes
import tensorflow as tf
# Create two tensors holding features and labels for MNIST
# 28x28 images, number is unknown (None)
x = tf.placeholder(tf.float32, [None, 784], name='Feature')  # name is useful in the GUI
# Create tensor to recognize 0-9 => 10 classes
y = tf.placeholder(tf.float32, [None, 10], name='Label')

# Values to compute for a linear classifier, for each pixel compute chance it is 0-9
W = tf.Variable(tf.zeros([784, 10]), name='Weights')
b = tf.Variable(tf.zeros([10]), name='Bias')

# Formulate solution with tensors
# Softmax returns a probability for each class that sums up to 1; here 10 classes
pred = tf.nn.softmax( tf.matmul(x, W) + b )
# Cost function for gradient descent; here is the cross entropy
cost = tf.reduce_mean(-tf.reduce_sum(y * tf.log(pred), reduction_indices=1))
# Learning algorithm to train the network
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Start execution, assume features and labels have been loaded
sess.run([optimizer, cost], feed_dict={x: features, y: labels})
TensorBoard [45]

- Browser visualization for Tensorflow
  - Model graph
    - Operations can be given names
    - Name scopes organize operations together
  - Results: scalar, events, histogram, multi-dimensional, audio, images, ...
  - e.g., visualize accuracy over training steps
  - Embeddings: a mapping from discrete objects, e.g., words, to vectors of real numbers
- Reads (binary) output from a log directory, can be multiple runs!
  - Uses protocol buffers for serialization
  - Analyze hyperparameter search
TensorBoard: Model Graph Visualization
TensorBoard: Result Visualization

TensorBoard

SCALARS

GRAPHS

INACTIVE

- Show data download links
- Ignore outliers in chart scaling
- Tooltip sorting method: nearest

Smoothing

Horizontal Axis

- STEP
- RELATIVE
- WALL

Runs

- Write a regex to filter runs

Logs/

accuracy

loss

accuracy

accuracy
The (Apache) Hadoop community is active

Software responsibilities:
- Hadoop deployment and cluster management
- Data management and provenance
- Security
- Analysis
- Automation (scheduling, data ingestion)

One goal: simple usage
- Alternative user interfaces
- Research of domain-specific languages (XML based or language embedded)

Many software packages are used but still in Apache incubator (beta)
Bibliography

10 Wikipedia
11 http://hortonworks.com/blog/introduction-apache-falcon-hadoop/
12 http://gethue.com/
13 http://falcon.apache.org/
14 http://falcon.apache.org/0.7/EntitySpecification.html
15 http://oozie.apache.org/
16 http://oozie.apache.org/docs/4.2.0/index.html
17 https://github.com/yahoo/oozie/wiki/Oozie-Coord-Use-Cases
18 http://sqoop.apache.org/
19 http://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html
20 http://slider.incubator.apache.org/
21 http://slider.incubator.apache.org/docs/slider_specs/application_package.html
22 https://knox.apache.org/
23 http://hortonworks.com/blog/apache-atlas-project-proposed-for-hadoop-governance/
24 http://ranger.incubator.apache.org/
27 https://github.com/cloudera/impala
29 http://incubator.apache.org/drill/
30 https://drill.apache.org/docs/json-data-model/
31 http://lucene.apache.org/solr/features.html
32 http://lucene.apache.org/solr/quickstart.html
33 http://solr-vs-elasticsearch.com/
34 http://mahout.apache.org/
35 http://www.scala-lang.org/
36 http://h2o.ai/
37 https://mahout.apache.org/users/recommender/quickstart.html
39 https://zeppelin.incubator.apache.org/
40 http://hortonworks.com/products/data-center/hdp/
41 http://hortonworks.com/apache/kafka/
42 http://kafka.apache.org/intro
43 http://metamodel.apache.org/
44 https://www.tensorflow.org/
45 https://github.com/tensorflow/tensorboard
46 https://www.tensorflow.org/get_started/mnist/beginners